Chapter 11
Upwind, Counter-Clockwise, 4-bladed Turbine
(Example Case 5)

[NOTE: This chapter reproduces some of the information in Chapter 10 - Example Case 4, but is meant as an adavanced example to show how ADAMS/WT can create upwind and counter-clockwise rotating turbines. The user is expected to have gone through a few of the other examples and not need the level of detail found in previous examples. Also, this is not supposed to be a realistic design, since most of the components are sized from the 2-bladed examples. This is very clear in the rotor's response during the simulation!]

To demonstrate the entire process, this section describes the construction, in ADAMS/WT, of an example 4-bladed, upwind, horizontal-axis machine. The construction steps are listed below, then discussed in more detail in the following text. We assume that you have already placed the *wt20.exe, wtblade.exe* and *wttower.exe* programs in the working directory.

11.1 Outline

NOTE: To work through this example, you should switch into the *nrel/examples/case_5* directory before starting ADAMS/View. You can then start View and load the ADAMS/WT overlay with the by reading in the command file *wt_main.cmd*. At this point you should be ready to begin *case_5*.

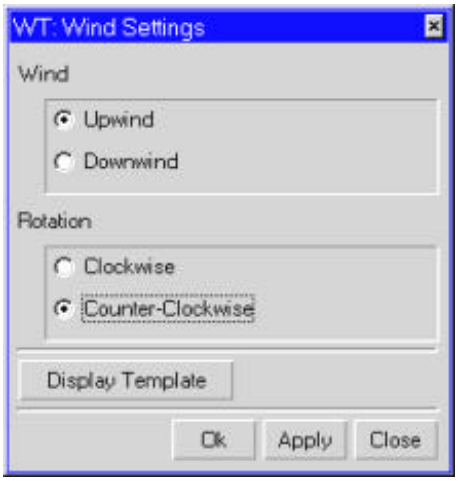
1. Set the direction of rotation and rotor orientation.
2. Create the tower aggregate element.
3. Create the nacelle aggregate element on top of tower.
4. Create the power train in the nacelle.
    5. Stator
    6. Low-speed Shaft
    7. Motor-Generator
8. Create the 4-bladed rigid hub aggregate element.
9. Relocate the hub to the end of the low-speed shaft.
10. Create blades #1, #2, #3 and #4.
11. Relocate blades to their attachment points on the hub.
12. Add AeroDyn aerodynamics to each blade.
13. Add gravity.
14. Add desired output requests.
15. Do the analysis.
16. Look at the results.

NOTE: In order to avoid losing your work, we recommend that you save the ADAMS/View session to a binary file after each section in the example is completed. This can be done through the FILE menu by Save Database As, or from the command line by typing:

file binary write file=case_5 (or just *fi bi wr fi=case_5*)

11.2 General Set-Up

This example uses the upwind, counter-clockwise set-up. From main WT menu, select WIND / ROTATION SET-UP and the set Upwind and counter-Clockwise from the available selections. The internal ADAMS/View variable *dir_rot* will be set to the string "UCC".



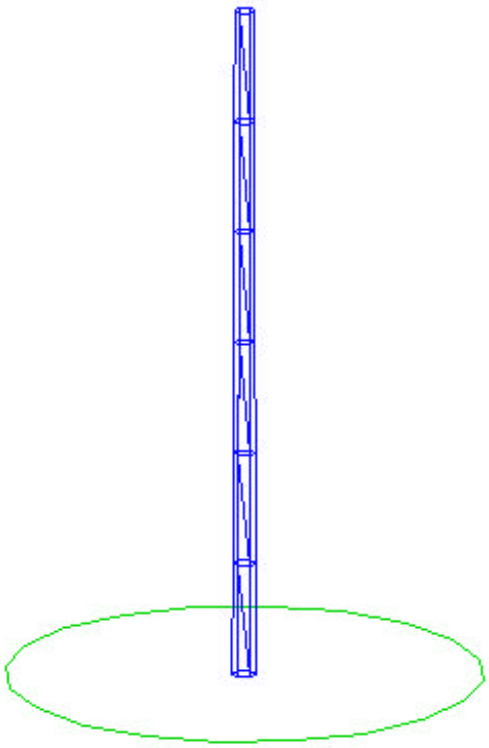
11.3 Tower

The input data for the case_5 tower are in *tower.dat* in the *examples/case_5* directory. These are the same as for the previous 4 examples. Bring up the tower create panel and use the following values:

Number of Parts = 6
Tower Height = 26 m
Tower Properties File = tower.dat
Number of Sides = 6
Bottom Diameter = 1.0 m
Top Diameter = 0.8 m
Color = your choice

If everything is working properly, ADAMS/WT should display an information window which monitors the automatic tower construction. Depending on the speed of your platform, building the tower may take several minutes. When the macro terminates, the info window should disappear and you should see the tower.

If you desire, you can add some graphics to the *ground* PART to give some perspective during the subsequent modeling. Here you can see the effect of adding a 10-m radius circle graphic centered on the *O* MARKER:



This would be a good place to save your work.

11.4 Nacelle

After completing the tower, we next move to the nacelle. In this example, the nacelle must face upwind. You can use the following data in the NACELLE CREATE panel:

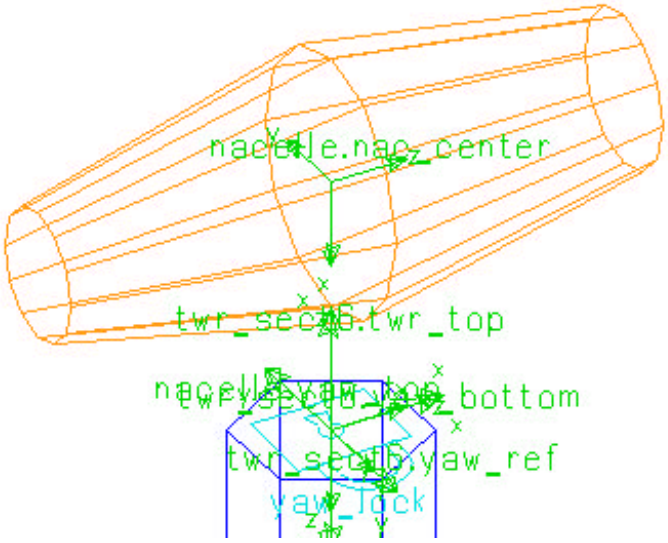
Yaw Type **= L**ocked
Marker_on_Tower = yaw_bottom
Shaft_Height_Above_Bearing = 1.0 m
Yaw Stiffness = 0
Yaw Damping = 0
Diameter_at_Bearing = 1 m
Upwind Length = 1.3 m
Upwind Diameter = 0.5 m
Downwind Length = 1.3 m
Downwind Diameter = 0.6 m

Then, bring up the NACELLE MODIFY panel to set the mass properties for the nacelle. When you hit the MASS PROPERTIES button, it will automatically bring up the standard View PART MODIFY panel for the *nacelle* PART. You should enter the following values:

mass = 4000 kg
center_of_mass_marker = nac_center
ixx = 1500 kg-m^2
iyy = 1500 kg-m^2
izz = 10 kg-m^2

After completing (with OK) the PART MODIFY panel, you should then **Close** out of the NACELLE MODIFY panel.

By picking on the nacelle and rotating a bit, you should see something like this:

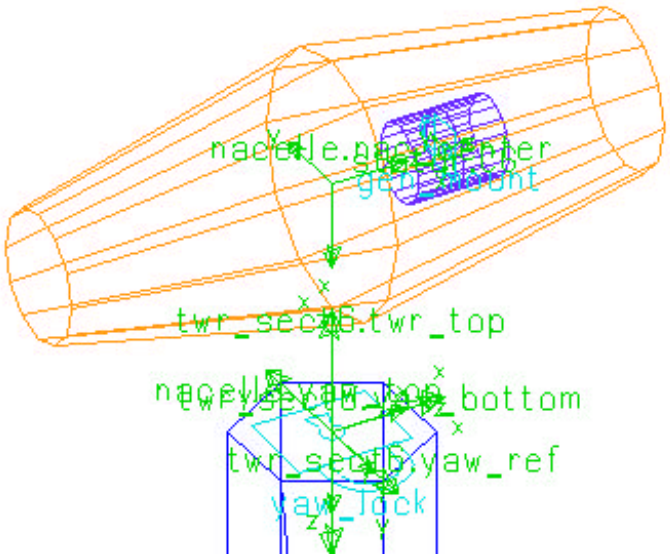


11.5 Power Train

Begin the power train by creating a generator body, called the *stator*. Remember that you can split the non-rotating inertia between the nacelle and the stator as you see fit. However, it is convenient to have a *stator* PART for connecting to one side of the motor-generator later on. Bring up the POWER_TRAIN menu, then select the GENERATOR STATOR panel and enter these values:

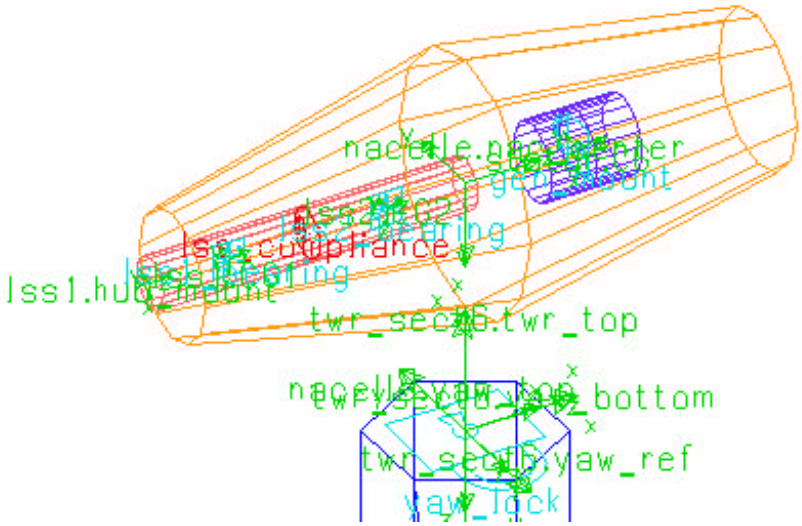
Location = 0,0,0.5 m
Relative_to = nac_center
Mass = 200 kg
I_xx = I_yy = 20 kg-m^2
I_zz = 10 kg-m^2
Graphics Diameter = 0.3 m
Graphics Length = 0.4 m

This should add the *stator* to the nacelle as shown here:



Next, select the POWER_TRAIN LOW-SPEED_SHAFT TORSION_ONLY panel and enter the following values:

Location = 0,0,-0.7 m
Relative_to = nac_center
Diameter = 0.2 m
Length = 1.4 m
Stiffness = 9.8E6
Damping = 9.8E4
Mass = 100 kg
Ixx (=Iyy) = 10 kg-m^2
Izz = 1 kg-m^2



Since there is no high-speed shaft or gearing, next you should create the motor-generator. Here, the voltage is stepped up smoothly from zero to 240 volts over the first 200 msec of the simulation. This allows for a static solution, and for a more gentle startup. Bring up the POWER_TRAIN MOTOR-GENERATOR panel and enter these values.

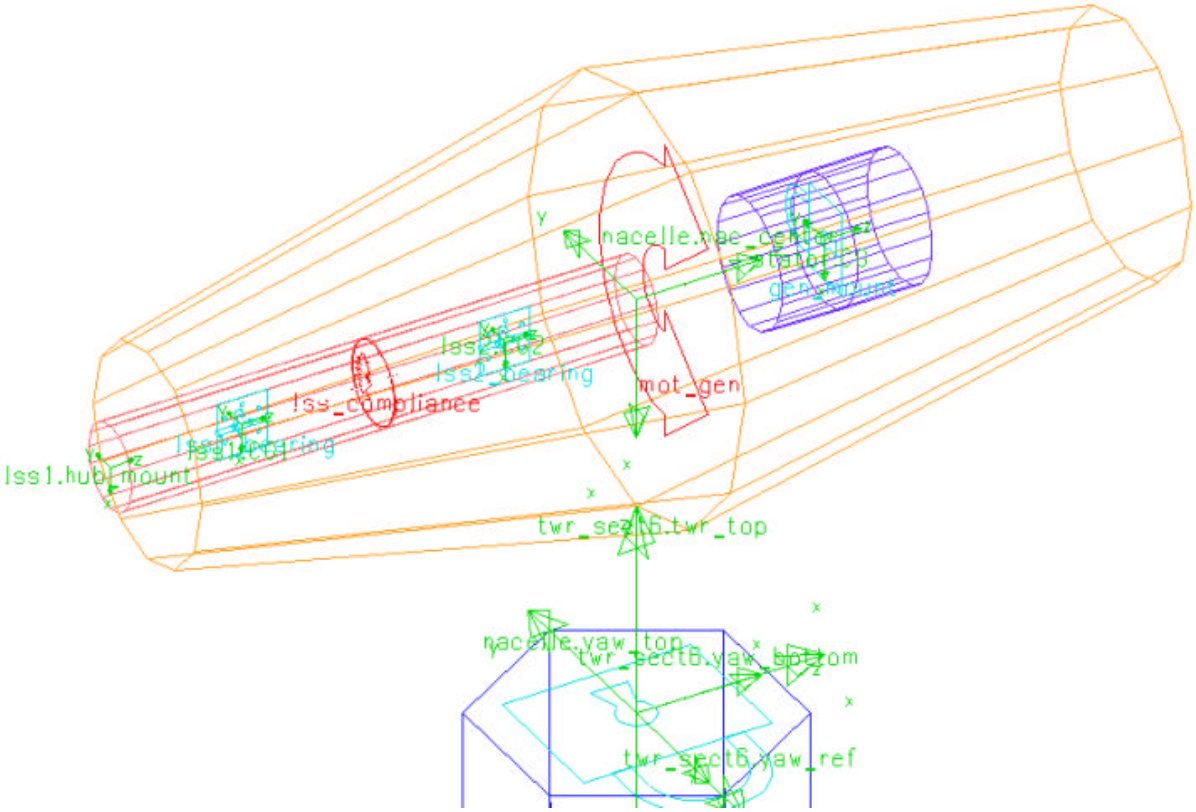
Line Voltage = STEP(TIME,0,0,0.2,240)
Desired Speed (rpm) = 60
I Marker = .hawt.lss2.CG2
J Marker = .hawt.stator.CG

The torque-voltage-speed relation in the TORQUE_FUNCTION entry is described for this case using Thevenin's equation, which is automatically generated by filling in these coefficients:

A_0 = 0.0128067
C_0 = 0.000157
C_1 = 0.00106
C_2 = 0.02428

and hitting the Generate Torque Function button to load the correct expression in the TORQUE_FUNCTION field.

Executing the motor-generator panel (OK) will create the *mot_gen* rotational SFORCE element which acts like the real motor-generator. The completed power train looks like:



11.6 Hub

ADAMS/WT has four options for rotor hubs, 2-bladed teetering and 3-, 4- or 5-bladed rigid. For any of the hubs, you can later add flexibility between the blade attachment and the hub itself using the HUB MODIFY panels. For this example, we will create a 4-bladed, rigid hub using the ROTOR_HUB CREATE 4-BLADED_RIGID panel. When you select this option, WT will display a template which makes it easier to visualize exactly to what the various parameters refer.

You should create a 4-bladed rigid hub using the following parameters:

Precone = 5 deg
Axial_Offset = 0.0 m
Hub_Radius = 0.5 m

This will build the *hub* PART geometry at the global origin (base of the tower).

After creating the *hub* PART, you need to bring up the ROTOR_HUB MODIFY 4-BLADED_RIGID panel, hit the MASS_PROPERTIES button and enter:

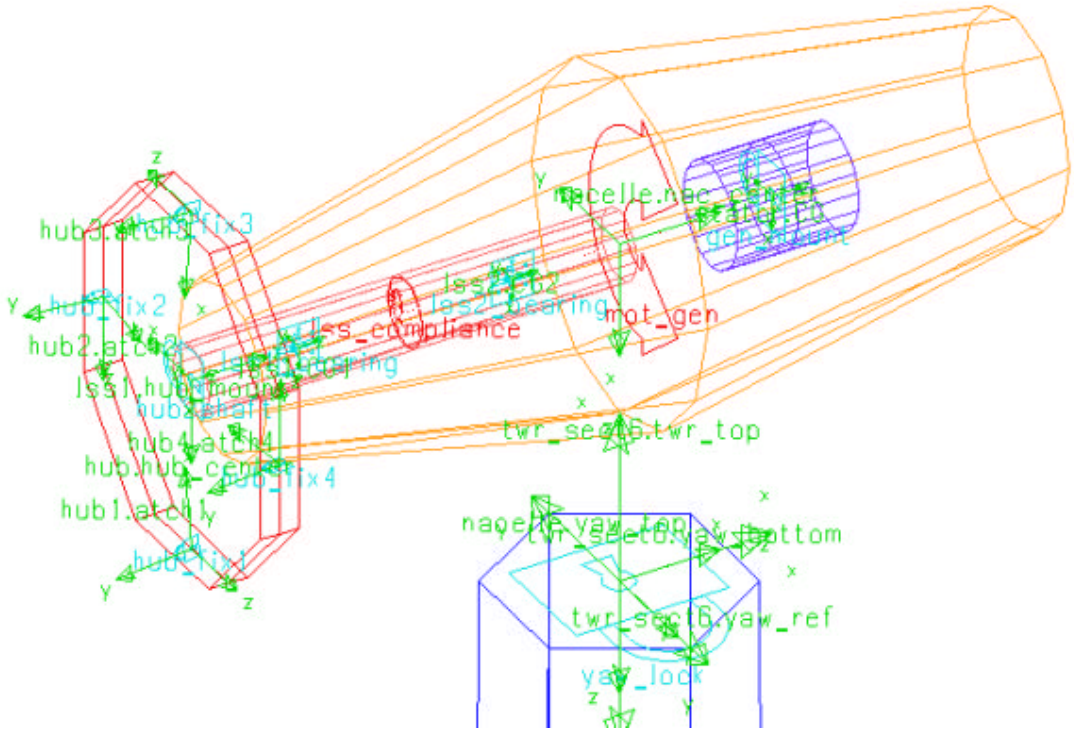
part_name = hub
mass = 200 kg
center_of_mass_marker = hub_center
ixx = 5 kg-m^2
iyy = izz = 30 kg-m^2

After updating the mass properties by hitting OK, you should **Cancel** out of the ROTOR_HUB MODIFY panel.

Next, you need to relocate the *hub* to the end of the low-speed shaft, using the ROTOR_HUB RELOCATE panel with these entries:

Base Marker on Hub = hub_center
Target Location Marker = hub_mount
Type of Attachment to Shaft = rigid

This will both move the *hub* to its correct position and create a fixed JOINT called *hub2shaft* between it and the *lss1* PART at the correct position and orientation. You may need to re-size the *hub2shaft* icon for convenience. You should then see something like:



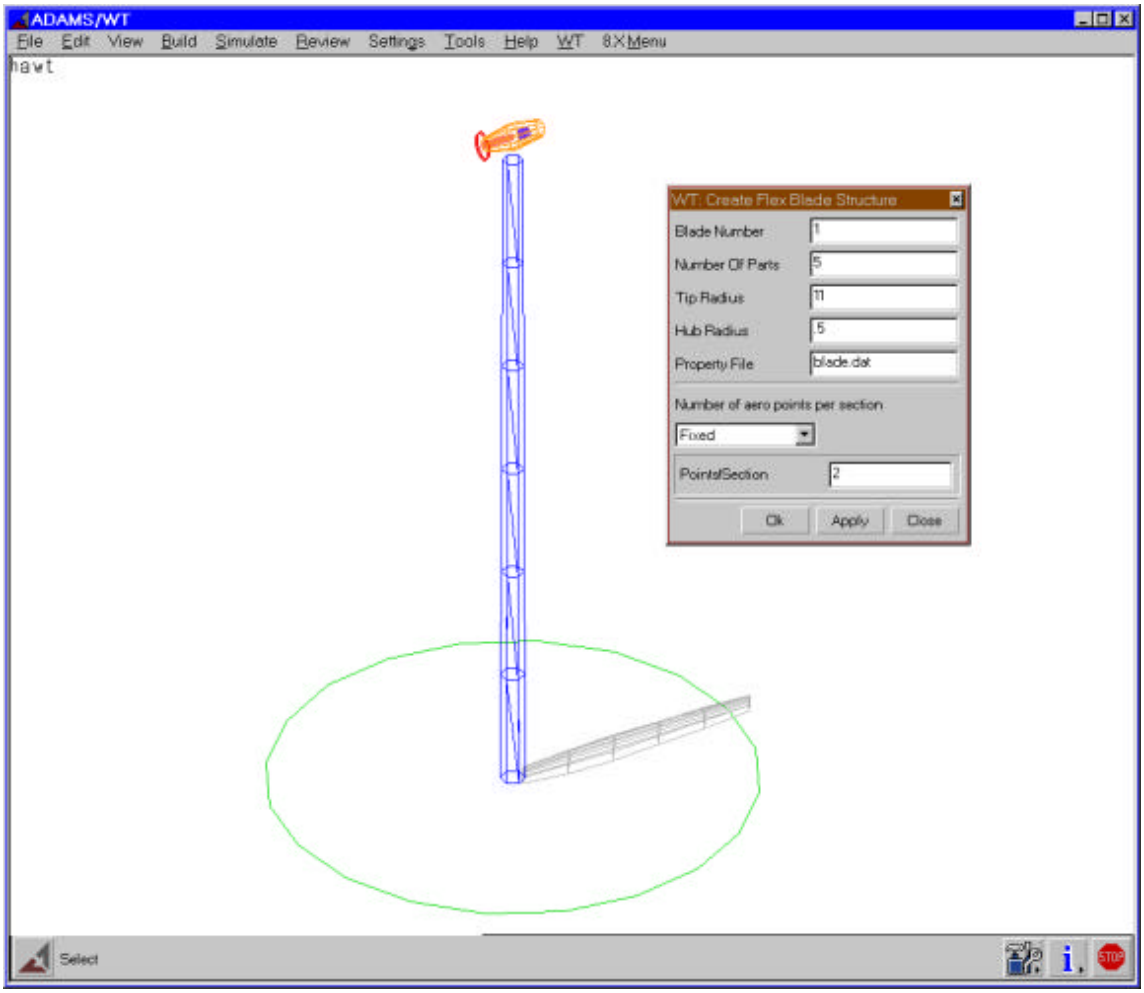
NOTE: This would be a good time to save your work.

11.7 Creating Rotor Blades

This example uses a fully flexible blade. The blade data for this case is found in the *blade.dat* file in the *examples/case_5* directory and is the same as for the previous examples. We will use a shorter blade, however. Bring up the flexible blade creation panel, ROTOR_BLADE CREATE FLEXIBLE_BLADE STRUCTURAL. You should use the following values:

Blade Number = 1
Number of blade parts = 5
Tip Radius = 11.0 m
Hub Radius = 0.5 m
File of blade properties = blade.dat
Fixed # of Aero_Points per Section = 2

When you APPLY the panel, ADAMS/WT will run the auxiliary program *wtblade*.exe and should display a window which monitors the blade construction, which could take a minute or so. The marker graphics have been turned off in the screen shown here.



If you APPLY the panel instead of OK ing it, you can create the second, third and fourth blades by just switching to the Blade Number field, entering 2, 3 or 4 and then selecting APPLY again. These blades will be built exactly on top of the first.

If you used OK on the first blade, you should again bring up the flexible blade creation panel through the menus, using ROTOR_BLADE CREATE FLEXIBLE_BLADE STRUCTURAL, and use the following values (WT should remember everything except the blade number.):

Blade Number = 2
Number of blade parts = 5
File of blade properties = blade.dat
Tip Radius =11.0 m
Tip Radius = 0.5 m
Fixed # of Aero_Points per Section = 2

Be sure to APPLY the panel this time.

Then, you can create the third and fourth blades by just switching to the Blade_Number field, entering 3 and 4 instead of 2, and then selecting APPLY. You will now have four blades right on top of each other at the base of the tower.

11.8 Relocating the Blades

After this, you need to relocate the blades to the correct attachment points on the hub. From the WT menu, use the ROTOR_BLADE RELOCATE panel with the following values. This will both move the first blade to the *atch1* MARKER on the hub and connect it to the hub with a half-length FIELD called *bl1_tbe0*.

Base Marker on Blade = bl1_root
Target Location Marker = atch1
Pitch Angle = -10.8 deg

Use the ROTOR_BLADE RELOCATE panel again with the following values to move the second blade to the *atch2* MARKER on the hub and connect it to the hub with a half-length FIELD called *bl2_tbe0*.

Base Marker on Blade = bl2_root
Target Location Marker = atch2
Pitch Angle = -10.8 deg

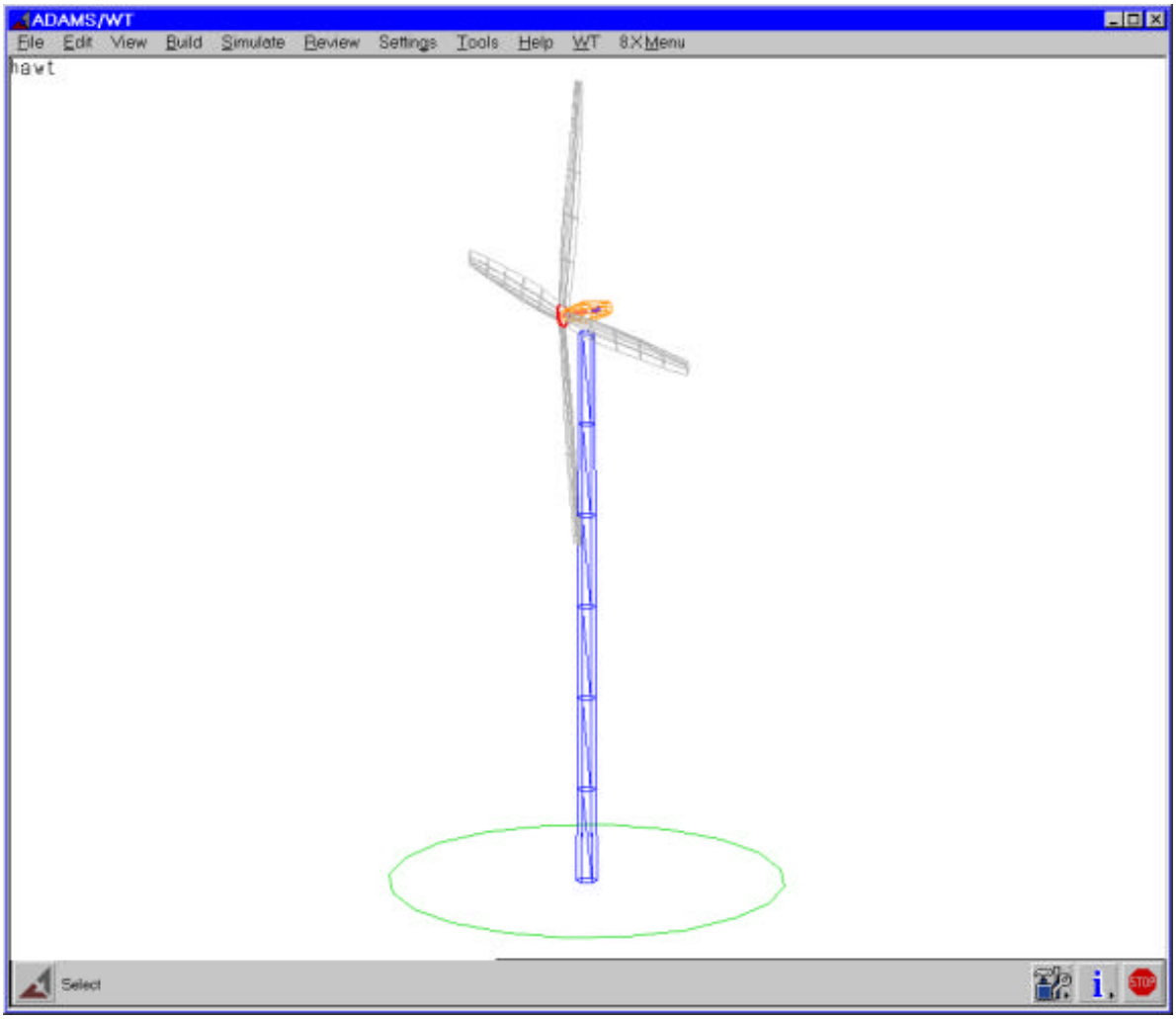
Use the ROTOR_BLADE RELOCATE panel again with the following values to move the third blade to the *atch3* MARKER on the hub and connect it to the hub with a half-length FIELD called *bl3_tbe0*.

Base Marker on Blade = bl3_root
Target Location Marker = atch3
Pitch Angle = -10.8 deg

Use the ROTOR_BLADE RELOCATE panel again with the following values to move the fourth blade to the *atch4* MARKER on the hub and connect it to the hub with a half-length FIELD called *bl4_tbe0*.

Base Marker on Blade = bl4_root
Target Location Marker = atch4
Pitch Angle = -10.8 deg

Note that, by convention, the #1 blade begins in the zero azimuth position, which is pointing straight down. The following graphic has the ADAMS/View icons turned off for clarity.



11.9 Aerodynamics

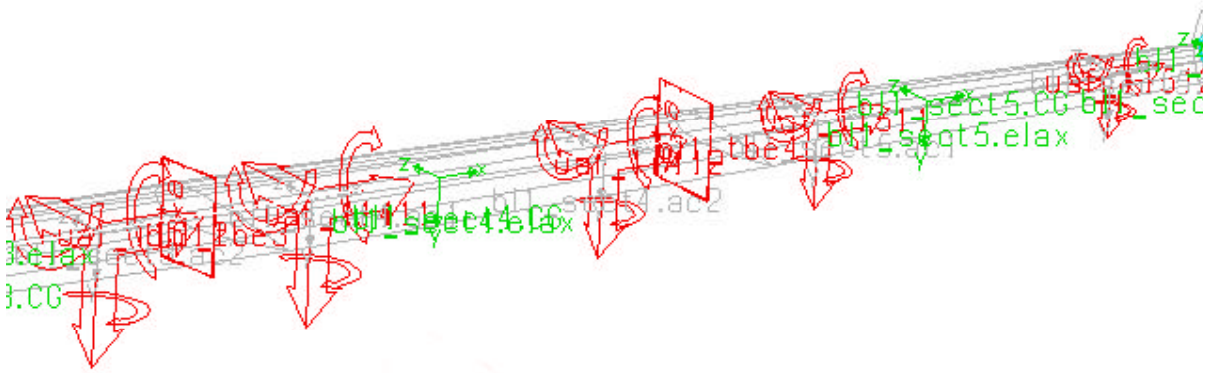
It is again simple to add aerodynamics to the model, due mainly to the large amount of up-front work done by Craig Hansen's group at the University of Utah, and the automation provided by ADAMS/WT. The AeroDyn aerodynamics subroutines are described in more detail in Appendix H. Remember, this version of WT is designed to work with version 10.0 of AeroDyn and will not work with previous versions.

To add the GFORCE elements which apply the aerodynamic forces computed in the AeroDyn routines to blade #1, you should bring up the AERODYNAMICS AERODYN_AERO FLEXIBLE_BLADE panel and enter the following data:

Blade_Number = 1
Number_of_Sections = 5

APPLY this panel for blade #1. Then change only the Blade_Number field to 2, 3 and 4 and APPLY the panel again for each additional blade.

Each time you submit the panel, A/View will spend some time doing computations. After the aerodynamic GFORCE elements are added, it will be nearly impossible to make out anything on the whole model when the View icons are turned on. By itself, a section of blade #2 would look like:



Before you can use the aerodynamics, you must add four specific MARKERs to the one of the low-speed shaft PARTs which AeroDyn will use to identify the blade azimuthal position. These MARKERs must have the ADAMS/Solver identifiers of 305#, where # is the blade number. Also they must be aligned such that their z-axes are along the shaft axis of rotation and their x-axes point radially outward in the plane formed by the shaft and matching blade axes.

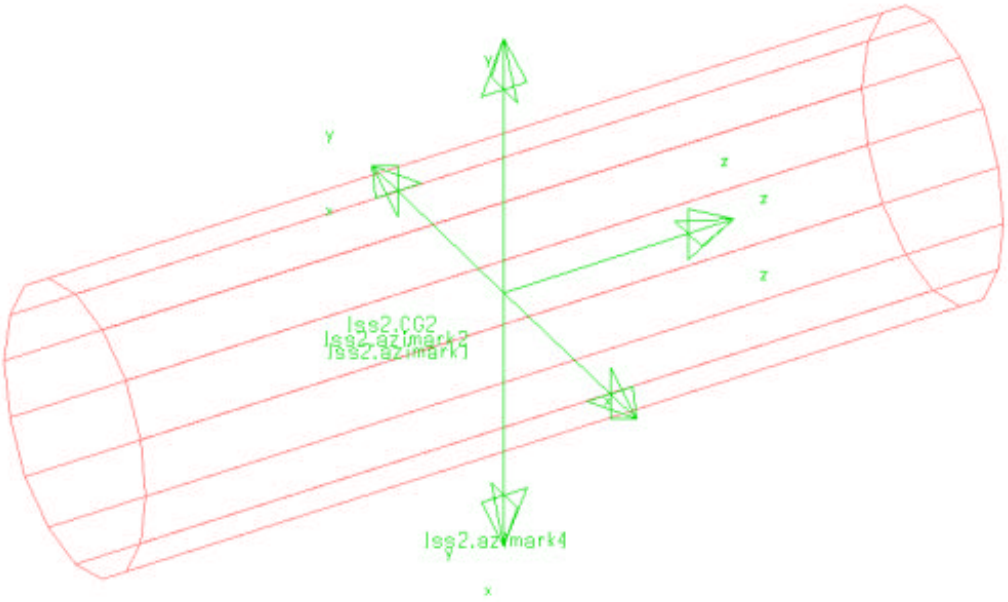
To do the first marker, you can go through the 8.X Menu to open the MARKER CREATE panel, or just use the WT-Specific Elements menu, then enter:

```
marker_name = .hawt.lss2.azimark1
adams_id = 3051
location = 0,0,0
orientation = 0,0,0
relative_to = CG2
```

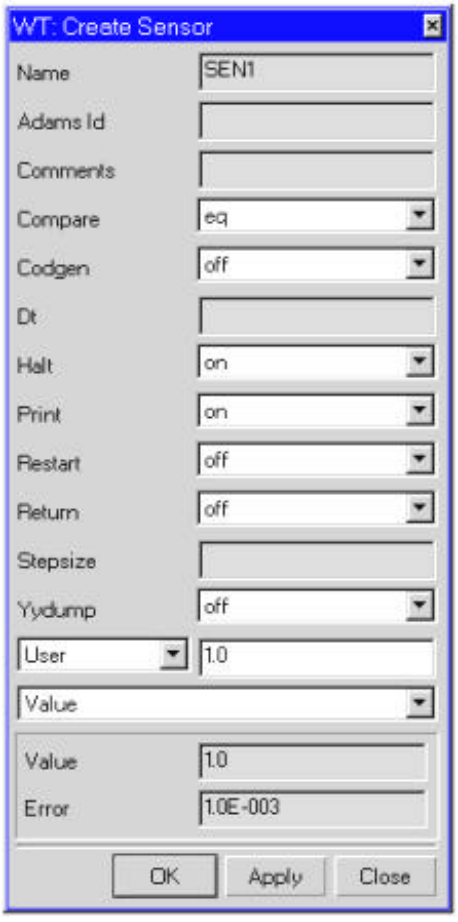
APPLY this panel, then re-do it as shown to create the markers for the other blades:

```
marker_name = .hawt.lss2.azimark2
adams_id = 3052
location = 0,0,0
orientation = 0,0,90
relative_to = CG2
and
marker_name = .hawt.lss2.azimark3
adams_id = 3053
location = 0,0,0
orientation = 0,0,180
relative_to = CG2
and
marker_name = .hawt.lss2.azimark4
adams_id = 3054
location = 0,0,0
orientation = 0,0,-90
relative_to = CG2
```

After creating these four MARKERs, you can display only the *lss2* PART and if you expanded the scale of the MARKERs slightly you should see:



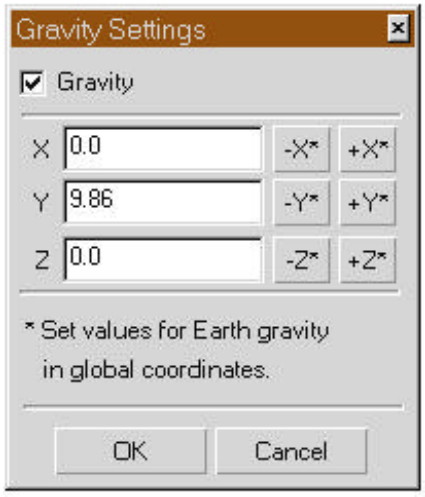
You must also create a dummy ADAMS SENSOR element to be used by the AeroDyn routines to monitor the progress of the simulation and advance the aerodynamic algorithm in step with it. This is also done through the WT-Specific Elements menu. This should bring up a SENSOR creation panel with all the correct values filled in for you. You can check it against this one.



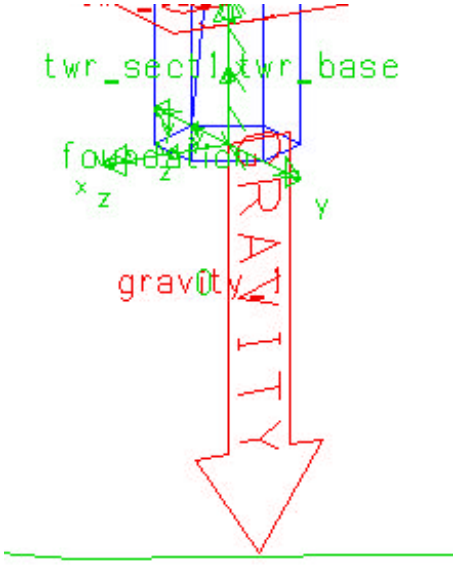
You will also need appropriate input files for AeroDyn. These are the *yawdyn.ipt*, *yawdyn.wnd* and *airfoil.dat* files which can be found in the *examples/case_5* directory. Note that *yawdyn.ipt* is not the same as for previous examples. Besides specifying four shorter blades, it also specifies a 0 tower shadow deficit becuase the rotor is upwind. Finally, you must have a user-executable version of ADAMS/Solver which includes the AeroDyn routines in order to run the model with these aerodynamics. You can copy this over from a previous example.

11.10 Gravity

Gravity should be added to the model using the normal ADAMS/View menus (Settings / Gravity) or from the WT-Specific Extras menu, which also has a gravity choice. Complete the panel as shown:



The gravity force should appear as a small arrow at the base of the tower, pointing straight down (along positive global y axis).



11.11 Output Requests

Basic requests for tabular output are now automated in ADAMS/WT version 2.0. To turn on these outputs, which will show up in the *.req* file, go back to the main WT menu and select REQUESTS to see the various different kinds of output you can solicit from ADAMS, most of which only need a confirmation to be included.

For the MOTOR-GENERATOR request, you must define the two MARKERs between which the *mot_gen* force acts. For this example, the WT defaults are correct.

M-G attachment marker on stator = .hawt.stator.CG
M-G attachment marker on LSS = .hawt.lss2.CG2

For the LOW-SPEED-SHAFT request, you must specify the *lss_compliance* torsional spring at the center of the low-speed shaft.

Which compliance element in LSS = lss_compliance

For the BLADE_ROOT request, you must specify which blade you want to monitor.

Which blade? = 1

The HUB_TEETER request, of course, is not used in this example.

You may, of course, add additional REQUESTs to the model using any and all of the methods allowed by ADAMS, i.e. standard type requests, functionally defined requests or REQSUB user-subroutine-generated requests. AeroDyn comes with an example REQSUB user-subroutine which is described in detail in the AeroDyn appendix.

As in previous versions of ADAMS/WT, to save both space and run time, we recommend that you use the standard WT-Specific Extras menu to turn off the RESULTS output. This minimizes the absolute amount of output and speeds the runs significantly. If you do need to use the results (.res) file, writing it UNFORMATTED will be both faster and produce much smaller files than the default FORMATTED output, but files will not be portable across platforms.

To turn off the results completely, set:

create_results_file = off
formatted = off

To keep the results file, but use the faster unformatted output, on the same panel you should set :

create_results_file = on
formatted = off

11.12 Doing the Analysis

The ADAMS/Solver output (*.out*) file also is normally superfluous, since the information there is a reformatted version of the request file. You can reduce the size of the *.out* file to a minimum from the main WT menu by selecting the panel for General Output Controls and setting:

print = off

You do not need to change any of the other fields in this panel. Nothing appears in your A/View model when this panel is executed; the ADAMS dataset will have a line in it which reads, "OUTPUT/NOPRINT."

In the past, it has often been necessary to "tweak" the static and dynamic solution parameters a little to get a wind turbine model to run the most smoothly. The integrators in ADAMS/Solver version 9.1, however, have been under continuous development for some time, as have the AeroDyn aerodynamics subroutines, and you should find that the combination in WT 2.0 works much better with default integrator and solver values than previously.

In addition, the numerical performance of the AeroDyn subroutines has been significantly improved. Because of this, it is now possible to run most WT-created models using the standard, default GSTIFF integrator. We recommend WSTIFF, however, because it appears to run more smoothly. Despite this appearance, GSTIFF may still run much faster. You should experiment with the integrators with your model.

To specifically select WSTIFF, from the Command Navigator, select the panel for EXECUTIVE_CONTROL SET NUMERICAL_INTEGRATOR_PARAMETERS. Cycle the Integrator Type to WSTIFF and execute the panel without changing any of the other parameters. This will put the INTEGRATOR/WSTIFF command right in the dataset and you should not need to put it in your Solver command file.

Using GSTIFF instead of the BDF integrators could make your simulations run significantly faster, possibly as much as 3-4 times faster. You may, however, see many more informational messages about integrator and solver performance "hiccups" than previously appeared using WSTIFF. (By default, the WSTIFF integrator neglects to tell the user about most of these little problems!) As long as the integrator continues and does not get bogged down, these messages can be considered just educational. To specifically select GSTIFF, from the Command Navigator, select the panel for EXECUTIVE_CONTROL SET NUMERICAL_INTEGRATOR_PARAMETERS. Cycle the Integrator Type to GSTIFF and execute the panel without changing any of the other parameters (but be sure that interpolate=off).

At this point, your model is complete and you should change its name from the default *hawt* to *case_5* and then write it out in dataset (*.adm*) format for simulation and in View binary and command file formats for safekeeping. These actions may be accomplished from the BUILD / MODEL / RENAME and FILE / EXPORT panels, or directly from the A/View command line. In the command line window, you can type:

model modify model=hawt new=case_5
file adams write file=case_5
file command write file=case_5 entity=case_5
file binary write file=case_5

Because you are running a user-executable version of ADAMS/Solver and will need special Solver commands to run it, and because you will often be running many simulations in a row, it is usually more convenient to run the code from the system command line instead of submitting it directly from the ADAMS/View. To do this you must first create an ADAMS/Solver command file *(.acf)* to control the simulation. Using your editor, create a text file named *case_5.acf* with the following contents:

```
case_5
case_5
integrator/err=.01
sim/dyn,end=0.25,step=50
integrator/err=1e-3
sim/dyn,end=0.5,step=50
sim/dyn,end=2,dtout=.02
sim/dyn,end=4,dtout=.02
sim/dyn,end=6,dtout=.02
sim/dyn,end=8,dtout=.02
sim/dyn,end=10,dtout=.02
stop
```

To run the code you can again use the menu interface step by step, or enter the single long command at the system prompt:

mdi -c ru-u i wt20.exe case_5.acf exit

or for NT

mdi ru-u wt20.exe case_5.acf exit

At this point, ADAMS/Solver should start up and the simulation progress should be displayed on screen. You can expect some difficulty with simulation startup, and a lot of warning messages about corrector convergence during the run, but these can both be ignored as long as the simulation recovers. The program log is also written to the file *case_5.msg*. When the run is complete, you should be returned to the system prompt and the simulation results should be in the files *case_5.gra* and *case_5.req*. The *.msg* file is not reproduced for this example.

11.13 Visualizing the Results

At this point, you are ready to read the results of the simulation back into ADAMS/View to look at the responses. Switch back to the A/View window and either use the FILE IMPORT menu or enter at the View command line:

file analysis read file=case_5 model=case_5

It will take View a few moments to read in the data from the graphics (*case_5.gra*) and request (*case_5.req*) files. You can then animate the results and see how the rotor responded.

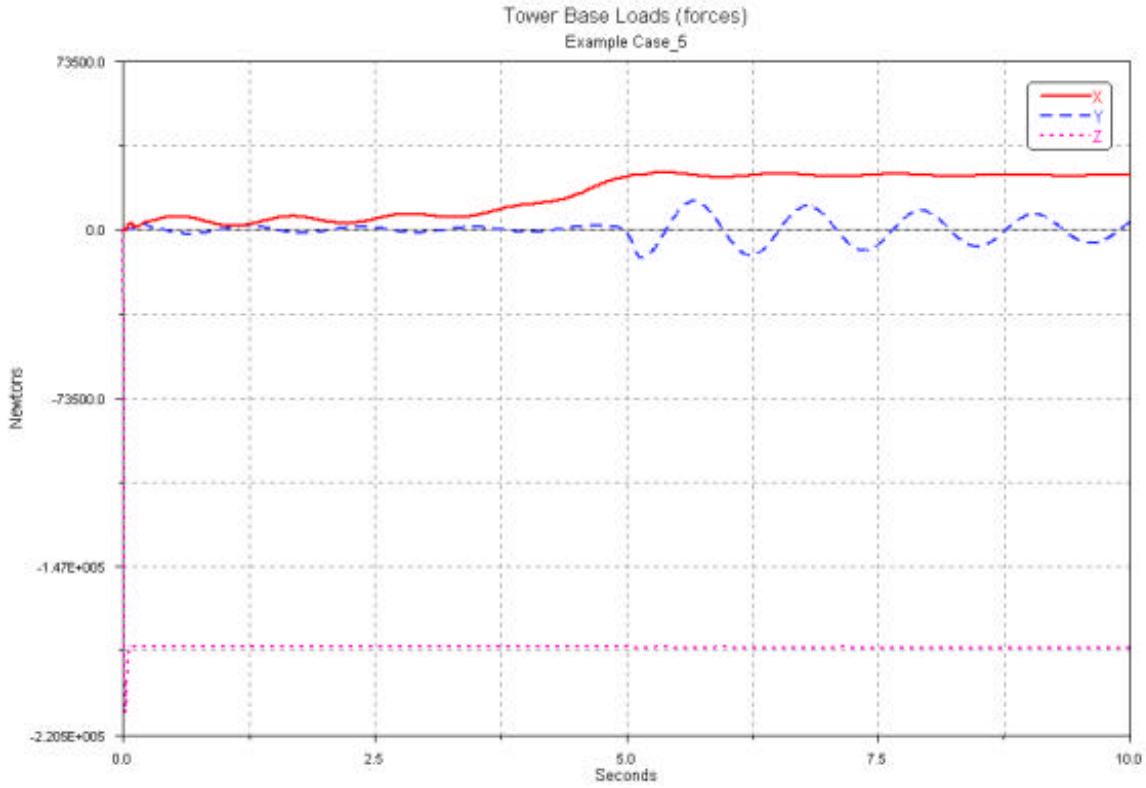
11.14 Plotting Output

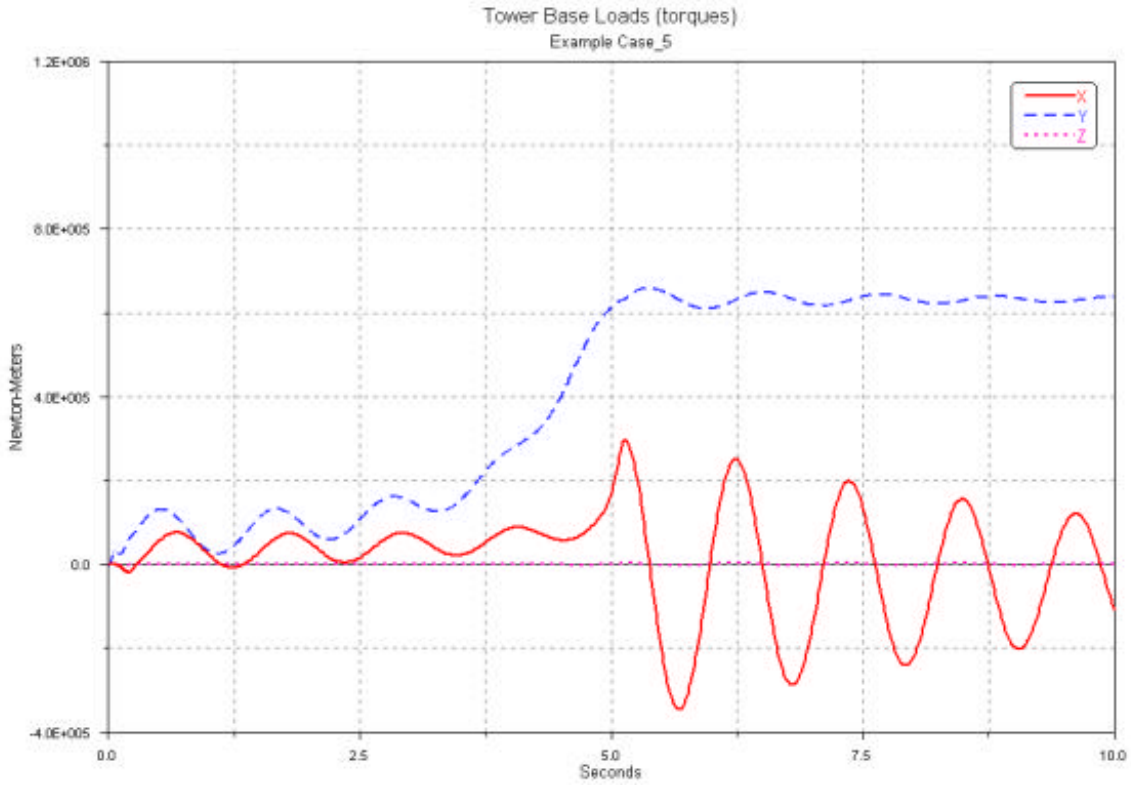
For repetetive plotting of specific requests from multiple simulations, it is often best to create a View command file (*.cmd*) containing the necessary commands to create and customize all the plots for a particular run. This command file contains the same commands you could type in at the View command line to create the plots, but is easily modifiable using a text editor for customization and changes. An example of such a command file is found in the file *plotemup.cmd* in the *examples/case_5* directory. The contents are not repeated here.

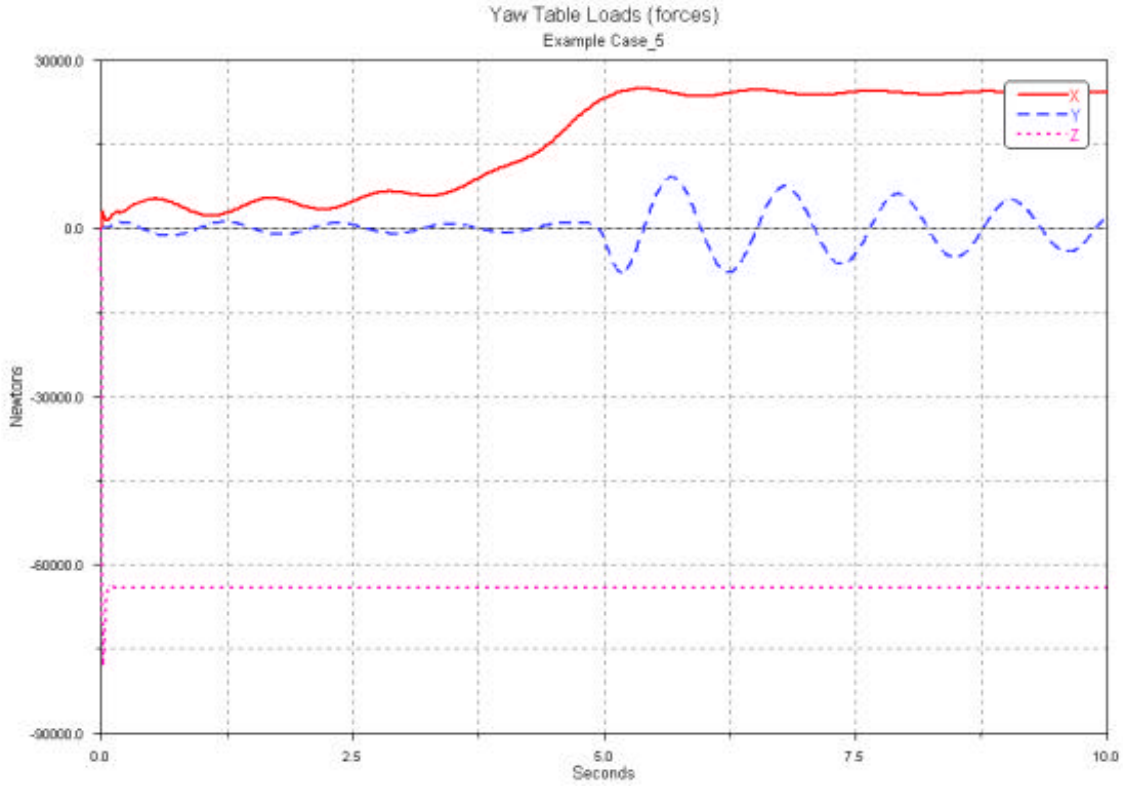
You can read in and run this command file through the FILE IMPORT menu or by entering at the View command line:

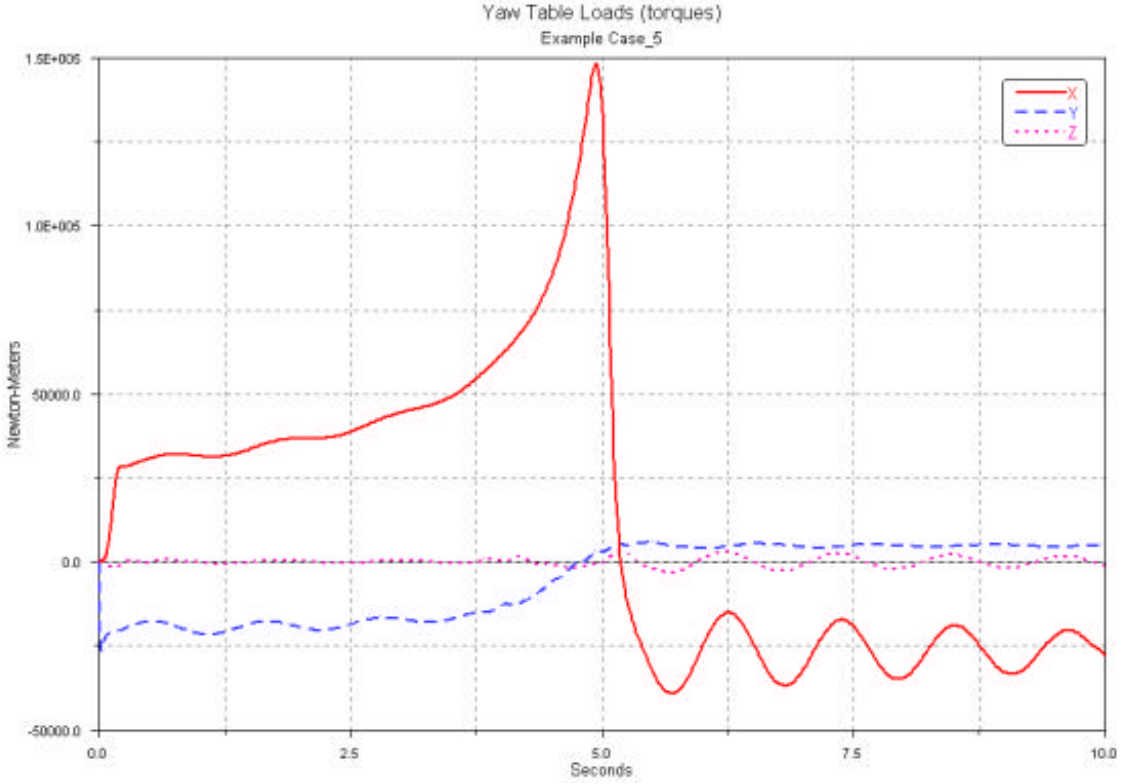
file command read file=plotemup

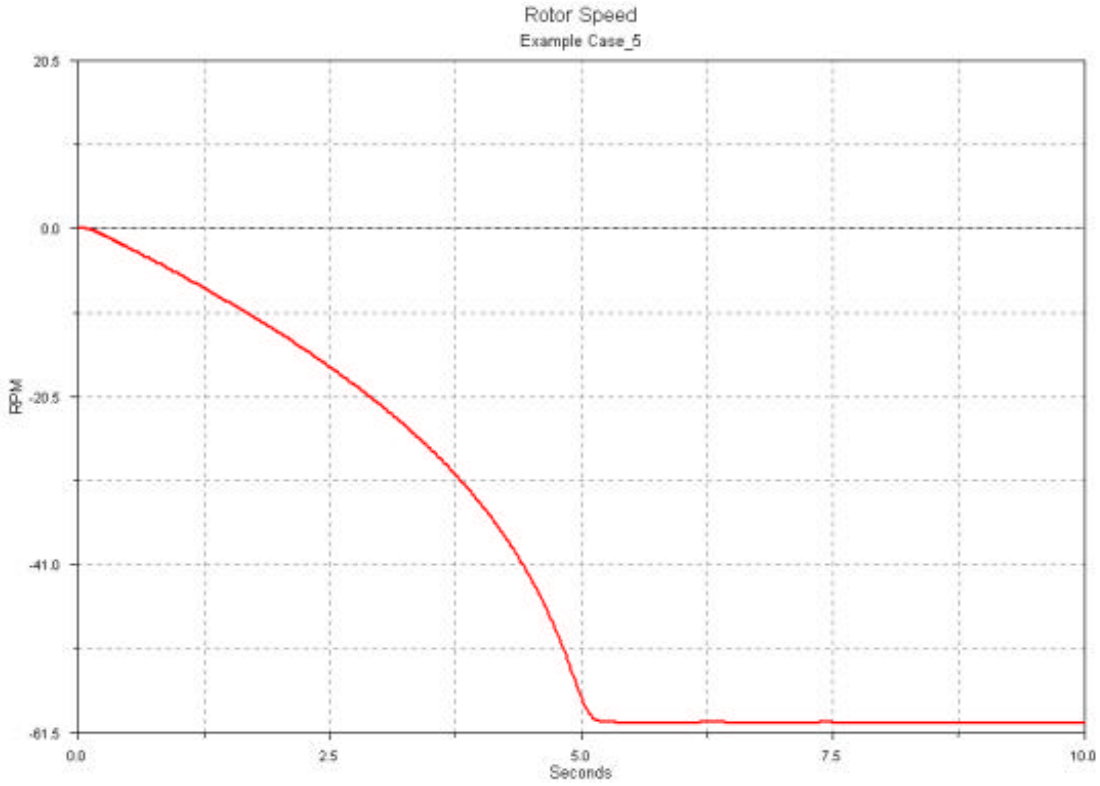
The example plots below can be used to confirm that your model and WT executable are working correctly.

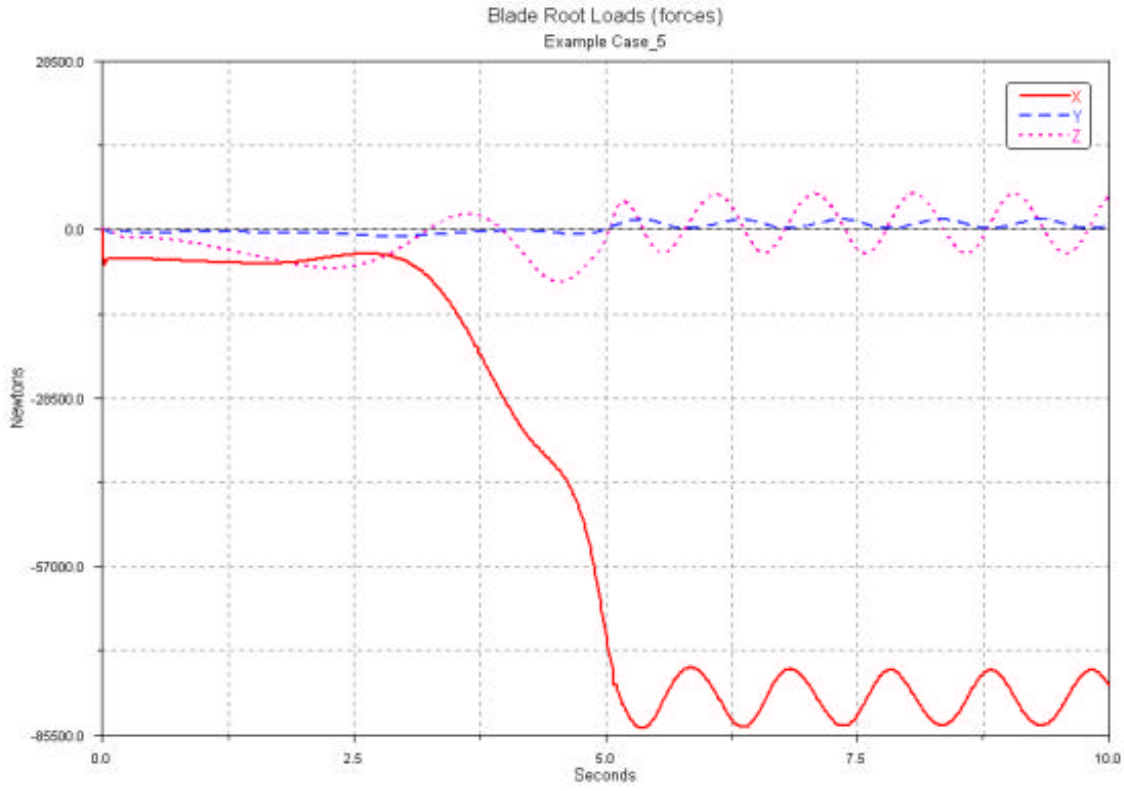

Tower Base Loads (forces)
Example Case_5
73500.0
0.0
-73500.0
-1.47E+005
-2.205E+005
Newtons
X
Y
Z
0.0
2.5
5.0
7.5
10.0
Seconds

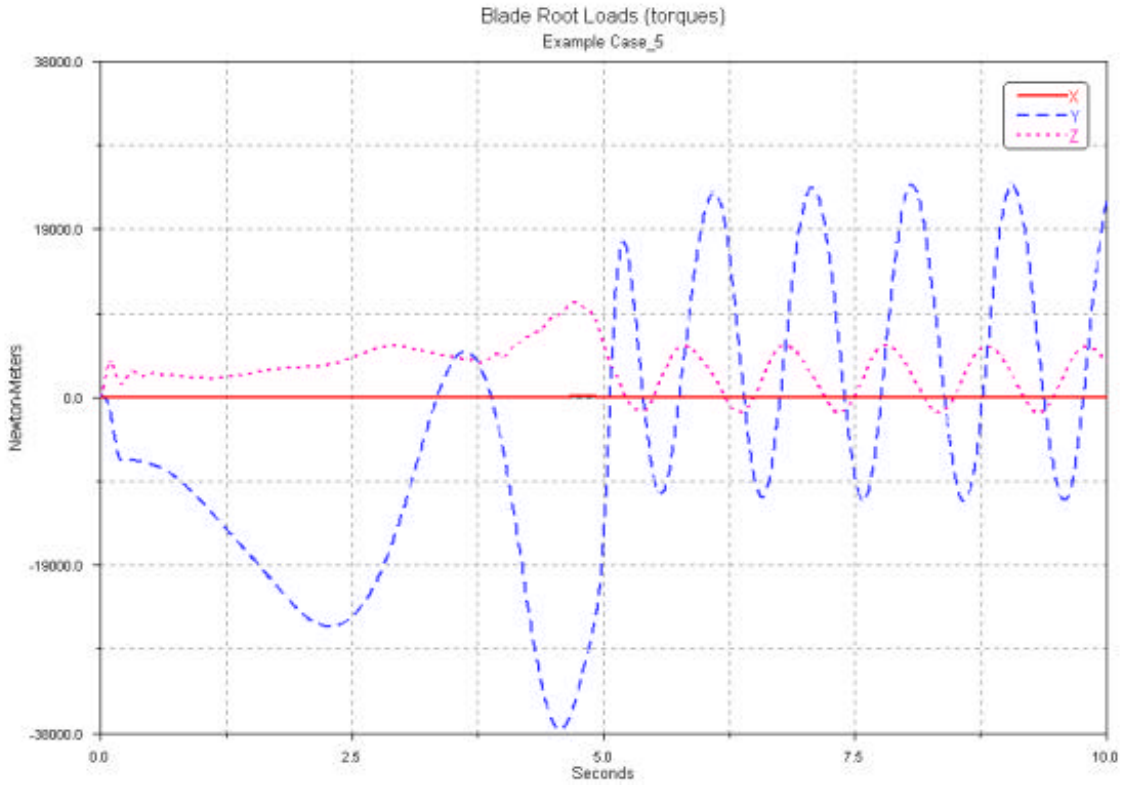

Tower Base Loads (torques)
Example Case_5
1.2E+006
8.0E+005
4.0E+005
0.0
-4.0E+005
Newton-Meters
X
Y
Z
0.0
2.5
5.0
7.5
10.0
Seconds

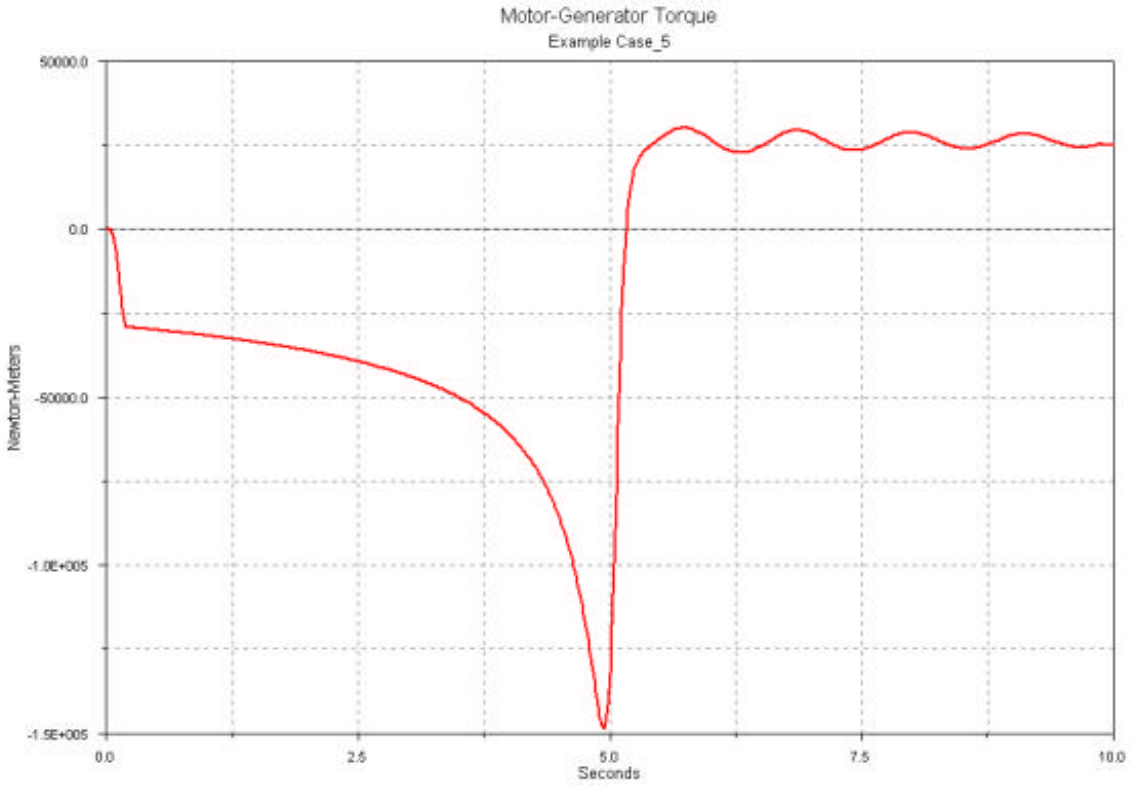

Yaw Table Loads (forces)
Example Case_5
30000.0
0.0
-30000.0
-60000.0
-90000.0
Newtons
X
Y
Z
0.0
2.5
5.0
7.5
10.0
Seconds

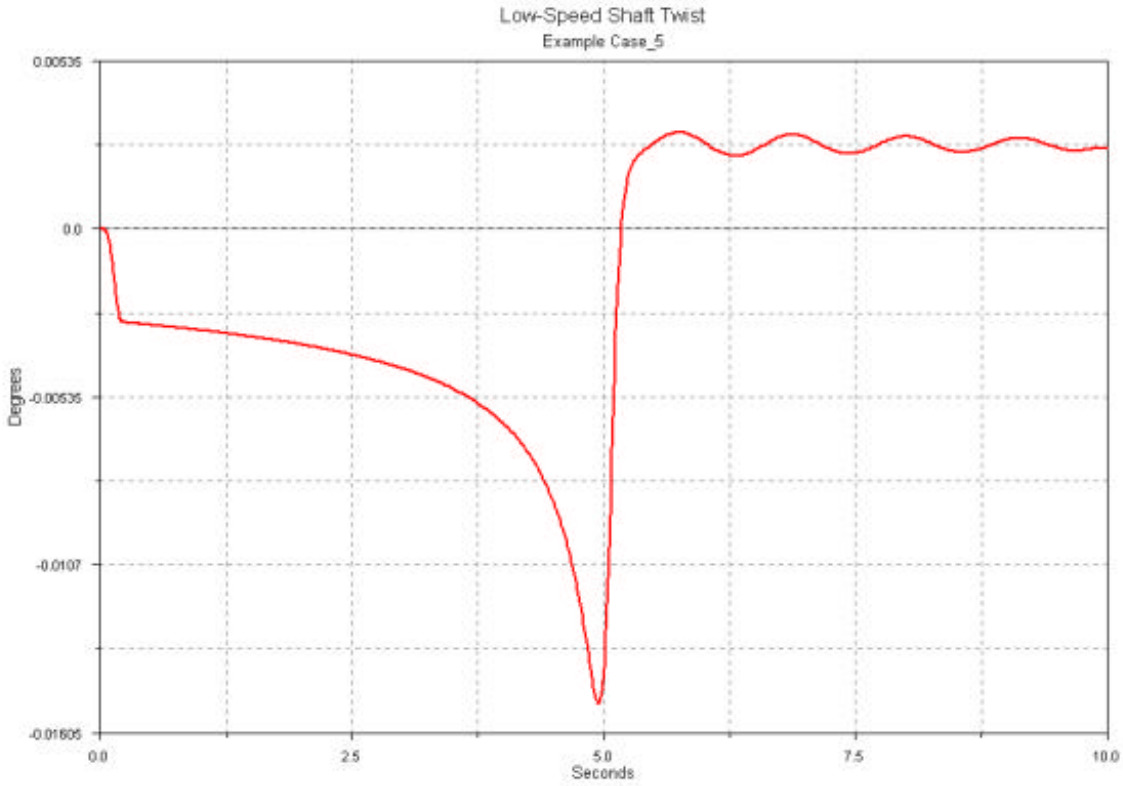

Yaw Table Loads (torques)
Example Case_5
1.5E+005
1.0E+005
50000.0
0.0
-50000.0
Newton-Meters
X
Y
Z
0.0
2.5
5.0
7.5
10.0
Seconds


Rotor Speed
Example Case_5
20.5
0.0
-20.5
-41.0
-61.5
RPM
0.0
2.5
5.0
7.5
10.0
Seconds


Blade Root Loads (forces)
Example Case_5
28500.0
0.0
-28500.0
-57000.0
-85500.0
Newtons
X
Y
Z
0.0
2.5
5.0
7.5
10.0
Seconds


Blade Root Loads (torques)
Example Case_5
38000.0
19000.0
0.0
-19000.0
-38000.0
Newton-Meters
X
Y
Z
0.0
2.5
5.0
7.5
10.0
Seconds


Motor-Generator Torque
Example Case_5
50000.0
0.0
-50000.0
-1.0E+005
-1.5E+005
Newton-Meters
0.0
2.5
5.0
7.5
10.0
Seconds


Low-Speed Shaft Twist
Example Case_5
0.00535
0.0
-0.00535
-0.0107
-0.01605
Degrees
0.0
2.5
5.0
7.5
10.0
Seconds